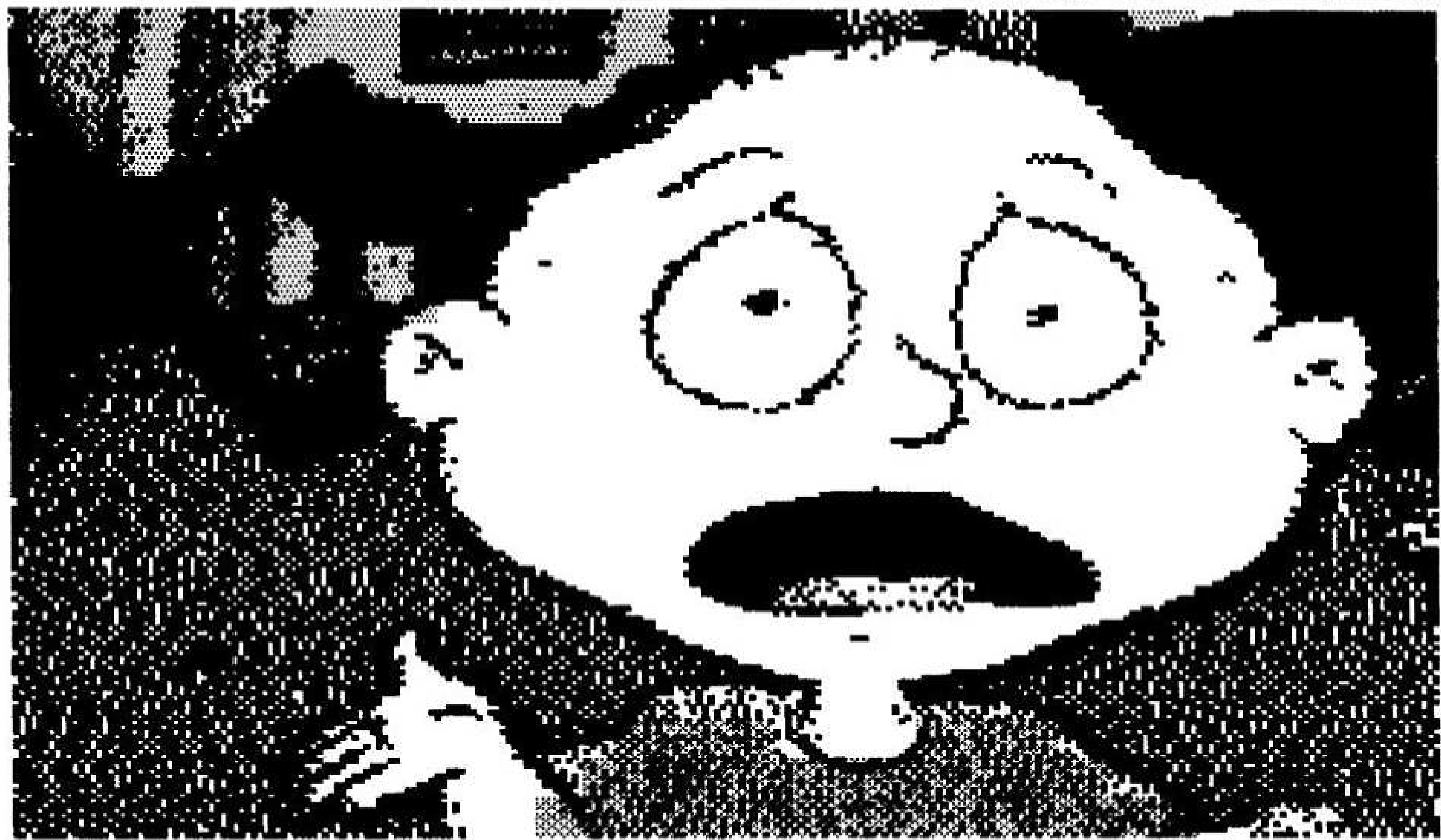
Spectrum Profi Club

für alle Spectrum und SAM Freunde



... diesmal mit herausnehmbarem Mitgliederverzeichnis!

Wolfgang & Monika Haller, Tel. 0221/685946
Penningsfelder Weg 98a, 51069 Köln
Bankverbindung: Dellbrücker Volksbank
BLZ 370 604 26, Konto-Nr. 7404 172 012

Ausgabe 87

März 1997

Shortcuts

Programmierwettbewerb
Kurz vor Redaktionsschluß rief uns Peter Meindl an und teilte uns mit, das im kommenden April-Info der Sieger bekannt gegeben wird. Insgesamt haben 4 User teilgenommen (es hätten ruhig ein paar mehr sein dürfen).

Gloucester Show
Die nächste "Spring '97 Spectrum & SAM Show" in der Village Hall, Quedgeley, Gloucester wirft ihre Schatten voraus. Als Datum wurde Samstag, der 19. April 97 festgelegt.

Let's go to Houten...
Zuvor findet aber am Samstag, dem 29. März wieder das Treffen in Houten mit Mitgliedern der SGG statt. Interessant für alle ZX81, Spectrum und SAM Freaks (auch Emulator). Ich hoffe auch diesmal wieder auf rege Beteiligung seitens unseres Clubs.

8Bit-Net
Vor einem Jahr wurde von einigen 8-Bit Computerfans das 8Bit-Net gegründet. Dieses bietet neben Echos für die verschiedenen 8-Bitter und Echos für systemübergreifende Themen auch ein Filenetz an, welches gerade im Aufbau ist.
Bisher gibt es Echos für den CPC und C64. Gesucht werden aber auch noch Atari, Sinclair und MSX-User, die gerne aktiv mitmachen (z.B. mit Artikeln).
Der SPC steht zusammen mit dem ZX-Team zumindest schon einmal in der Adressen- bzw. Kontaktliste. Wer an einer Zusammenarbeit interessiert ist, oder aber auch nur so mal Kontakt aufnehmen will, kann dies über folgendes "Headquarter" tun:

Transvision BBS, Hannover
Wolfgang Noistering (Fido 2:241/1115)

Internet-Adressen

In ihrem letzten Brief an mich hat das WoMo-Team mir vorgeschlagen, doch einmal ein paar Internet-Adressen und -pages zu sammeln. Deshalb habe ich mich an diesen Unirechner gesetzt, und unter viel Ächzen und Stöhnen Netscape geladen und mal gekuckt, was es so zu kucken gibt:

http://www.nvg.unit.no/sinclair/planet/intro.htm
Das is zugegebenermaßen die erste Adresse in der Internet Spectrumwelt. Dieses Archiv enthält Tonnen von Software und Infos. Wenn man sich nicht der WWW-Warterei aussetzen will, empfiehlt sich die FTP-Adresse: ftp.nvg.unit.no

http://osiris.sund.ac.uk/ ca4aba/snaps.html
Hier findet sich "Andy's Sinclair ZX-Spectrum Page", die hauptsächlich deshalb interessant ist, weil sie ein besser sortiertes Angebot an Snapshots liefert, als nvg.unit.no.

http://www.comlab.ox.ac.uk/oucl/users/ian.collier/Spectrum/indes.html
Das ist Ian Coliers Homepage, und hier findet sich sein Emulator "XZ 80".

http://www.jetman.demon.co.uk/speccy/index.html
"Jetman's Speccy Nostalgia Trip" mit einigen amüsanten Definitionen, dazu interesante Links und wichtige Dinge wie die comp.sys.sinclair-FAQ ("DON'T POST NO BLOODY BINARIES - D. Burke")

http://ds.dial.pipex.com/town/parade/no50/speccy.html
"Sinclair Spectrum: Most Wanted" ist eine weitere Sammlung von Snapshots, allerdings kan man hier auch nach speziellen Snaps fragen.

http://home.virtual-pc.com/isblpx/index.html
"The Spectrum Adventurer" ist natürlich was ganz besonderes für meinen Gusto. Hier finden sich Spiele und Infos für uns Grafik- und Joystickverächter zuhauf.

Und last not least:

http://www.nvg.unit.no.sinclair/img7/
Eine Webseite, die sich endlich mal weniger mit den Emulatoren als mit dem echten Boliden beschäftigt. Hier finden sich wieder viele Links, aber auch Dinge wie die Adressenlisten von Usergroups (einschließlich unseres Lieblings-computerclubs), PD-Bibliotheken etc.

Das wär's. Natürlich gibt es noch viele weitere Spectrumpages, aber als Einstieg sollte dies hier genügen. Die Links, die sich in den meisten dieser Pages finden, sollten es euch einfach machen, von hier aus alleine weiterzusuchen. Von Interesse ist vieleicht noch die ZX-Spectrum newsgroup: comp.sys.sinclair. Leider dreht sich die Diskussion in letzter Zeit hauptsächlich um die neuesten Spielesnaps, doch ich habe schon mehr als eine gute Information dort gefunden.

Nele Abels-Ludwig, Am Mühlgraben 4
35037 Marburg, Tel. 06421/210272

Auch unsere ZX 81 Freunde vom ZX-Team haben inzwischen eine eigene Homepage

http://home.t-online.de/home/p.liebert/zx-team.htm
Ein Verweis zu unserem SPC ist hier auch enthalten. Deshalb möchte ich diejenigen, die selber eine Homepage halten bitten, ebenfalls ein Link auf das ZX-81 Team einzubauen.
Da einige unserer Mitglieder auch Mitglieder im ZX-Team sind, hier die e-mail Adresse, unter der ihr Peter Liebert-Adelt direkt kontaktieren könnt: **p.liebert@t-online.de**

DIE SEITEN FÜR DEN SAM!

Batz'n'Ball Levelcodes

Die nachstehenden Levelcodes von "Batz'n'Balls", einem Arkanoid-ähnlichen Spiel, sandte uns Nico Kaiser aus Ilmenau:

Level	Code	Level	Code
001	ABOLISH	100	QUIBBLE
003	ACRONYM	101	QUIDDLE
004	ACOLYTE	102	QUONDAM
005	AFFABLE	103	RAUCOUS
006	ALIMONY	104	REGULAR
007	BAGGAGE	105	REUNION
008	BANSHEE	106	RIPPING
009	BEATING	107	ROMANCE
012	BRITAIN	108	RUBICON
013	CAESURA	109	SCRAPPY
019	DECAGON	110	SHEBANG
025	ECLOGUE	111	SQUALOR
061	KENTISH	112	STEALTH
062	KESTREL	113	SUBFUSC
063	KEYNOTE	114	SYRINGA
064	KIBBUTZ	115	TENANCY
065	KINETIC	116	THIMBLE
066	KNUCKLE	117	TONSURE
067	LACQUER	118	TREMOLO
068	LAMBAST	119	TRIBUNE
069	LAYETTE	120	TYPICAL
070	LICENSE	121	UNIFORM
071	LIMINAL	122	UNSLING
072	LUCENCY	123	UNSWORN
073	MANSION	124	UNTRIED
074	MESSAGE	125	UPRAISE
075	MAWSEED	126	UTENSIL
076	MISERLY	127	VAMPIRE
077	MORPHIA	128	VARMINT
078	MURRAIN	129	VENISON
079	NOSTRIL	130	VERMEIL
080	NOXIOUS	131	VERTIGE
081	NUMERAL	132	VIBRATE
082	NUCLEUS	133	WARFARE
083	NUPTIAL	134	WARTIME
084	NOSTRUM	135	WARSHIP
085	OCARINA	136	WARRIOR
086	OFFSIDE	137	WARRANT
087	ONEROUS	138	WHATNOT
088	OSMOSIS	139	YARDARM
089	OUTWALK	140	YIDDISH
090	OXONIAN	141	YOGHURT
091	PADLOCK	142	YOUNKER
092	PARABLE	143	YULELOG
093	PESSARY	144	YIELDER
094	PETTING	145	ZEALAND
095	PREVIEW	146	ZOOLOGY
096	PYJAMAS	147	ZIONISH
097	QUALIFY	148	ZEALOUS
098	QUARREL	149	ZAMBIAN
099	QUERIST	150	ZYMOTIC

Und noch ein Tip von Nico: Nach der Zeile 'LOAD "B'N'B.0" CODE' soll die Zeile 'POKE 32787,1' eingesetzt werden. Dieser POKE bewirkt unendliche Leben.

Software-Test

Beetle Mania (G. Bobker)
Bezugsquelle: Persona Softw., 31 Ashwood Drive, Brandlesholme, Bury, Lancs, BL8 1HF, England.
Wer vom Spectrum oder PC her 'Sokoban' kennt, der weiß, worum es sich bei "Beetle Mania" handelt. Es ist das sehr bekannte Spiel, wo ein Mann Kisten in bestimmte Räume und auf bestimmte Felder schieben muß (bei "Beetle Mania" übernimmt dies ein Käfer - aha! daher also der Name). Das hört sich einfacher an, als es ist, denn oft wird der Spieler durch andere, sehr ungünstig plazierte Kisten behindert. Außerdem darf man keine Kiste in eine Ecke schieben, denn dann bekommt man sie nicht mehr weg. Dazu eine Unzahl an Leveln, die den Tüftlern unter euch eine Menge Kopfzerbrechen bereiten wird.
Einziger Kritikpunkt aus meiner Sicht ist, das nur über die Pfeiltasten gespielt werden kann, außerdem vermisse ich jegliche Musik. Nun, dem einigermaßen geübten SAM User dürfte es jedoch nicht schwerfallen, sich irgendeine Hintergrundmusik in das nicht allzugut geschützte Programm einzubauen. Unsere Meinung drücken wir durch ein neues Rating-System aus (1 = mies, 2 = naja, 3 = durchschnittlich, 4 = schon besser, 5 = aber hallo und 6 = whow!):

Grafik: ☐☐☐☐ **Musik:** ☐
Steuerung: ☐☐☐ **Fun-Faktor:** ☐☐☐☐☐

Labyrinth (Steve Ekins)
Bezugsquelle: Jupiter Software, 2 Oswald Road, Rushden, Northants, N10 OLE, England
Neu ist die Spielidee dieses 1994 entstandenen Spiels sicher nicht: In einem Labyrinth muß man im Kampf gegen die Zeit Disketten einsammeln und sich nicht von den unvermeidlich Bösen erwischen lassen - man hat nur ein Leben. Hat man 18 Disketten eingesammelt, muß man auch noch fix den Ausgang erreichen.
Die Kritik ist ähnlich obiger: Steuerung ist nur per Joystick (oder für Masos per Tasten 6-0) möglich. An "Musik" gibt es nur ein paar FX-Effekte, so das unser Rating so aussieht:

Grafik: ☐☐☐☐ **Musik:** ☐
Steuerung: ☐☐☐☐ **Fun-Faktor:** ☐☐☐☐



Der Festplatten-aufbau

- Teil 2 -

Letztesmal waren wir mit Kopf 0, Spur 0, Sector 0 beschäftigt und da machen wir heute weiter. Wichtig ist, das man bei jedem Speichersystem, egal ob es der Hauptspeicher des Computers, einer Diskette oder einer Festsplatte ist, alle Informationen wiederfindet, nach denen etwas abgespeichert wurde. Hierzu dienen die Pfeile (Pointers in englisch, auch Zeiger genannt), z.B. dienen die Bytes 30-33 in Sektor 0 als 4-Byte Sektor-Nummer für das 'Free Space Chain', was wiederum bedeutet, das hier ein Pfeil auf die Informationen für das HDOS zeigt, in dem steht, wo und wieviel Platz frei ist. Fast alle Pfeile in HDOS werden als 4-Byte Sektor-Nummern (32 Bits) abgelegt, was bedeutet, das HDOS bis zu 4.294.967.295 Sektoren verwalten kann! Normalerweise enthalten die Bytes 30 bis 33 die Werte '01 00 00 00', und da das niedrigste Byte in 30 und das höchste in 33 steht, ergibt das Sektor '00000001' als 'Free space chain FSAM'. 'FSAM' ist die Abkürzung für 'File Sektor Allocation Map', dazu später mehr. Das HDOS muß aber nicht nur freie Sektoren finden können um neue Daten abzuspeichern, sondern es muß auch bereits gespeicherte Daten finden können. Es braucht ein Directory und die Bytes 34 bis 37 sind ein Pfeil zum ersten Directory Sektor. Sie enthalten die Werte '02 00 00 00', was folglich Sektor '00000002' ergibt.
Schauen wir uns zuerst Sektor '1' an (Kopf 0, Spur 0, Sektor 1). Dies ist ein 'Free Space' FSAM, was bedeutet, das alle Informationen, die das HDOS braucht um einen 'freien Platz' für neue Files und Directories zu finden, in diesem Sektor sichtbar sind. Sektor '1' sieht bei mir z.B. so aus:

Kopf 0, Spur 0, Sektor 1:

Byte	**0- 3**	**01 00 00 00 (Pfeil zu diesem FSAM)**
	4- 7	**00 00 00 00 (Pfeil zum nächsten FSAM)**
	8- 11	**00 00 00 00 (Pfeil zum letzten FSAM)**
	12	**00 (Dirty flag)**
	13-109	**(File information block 97 Bytes)**
	110-111	**(Current Run)**
	112-311	**(50x run start adress - jeder Eintrag 4 Bytes)**
	312-511	**(50x run end adress - jeder Eintag 4 Bytes)**

Die ersten 4 Bytes sagen nur, das diese 'FSAM' in Sektor '00000001' steht. Das HDOS ist so klug, das es weiß, das was es von Sektor 1 liest auch in Sektor 1 gespeichert ist. Es ist aber nicht so dumm wie es aussieht, wenn es diese Information im Hauptspeicher des SAM hat, dann weiß es immer durch diese 4 Bytes, von wo es diese gelesen hat. Der Pfeil zum nächsten FSAM ist '0', sodaßes keinen weiteren gibt. Dieser Sektor enthält alles, was benötigt wird, um den freien Platz auf der Festplatte zu identifizieren. Der Pfeil zum letzten FSAM zeigt ebenfalls auf '0', da es keinen letzten gibt, dies ist das erste. Ein komplexeres File, das sagen wir mal jedoch 3 FSAM Sektoren braucht, könnte z.B. ungefähr so aussehen:

Sektor	Dieses	Nächstes	Letztes
00000010	10000000	21000000	00000000
00000021	21000000	22000000	10000000
00000022	22000000	00000000	21000000

Durch dieses System weiß das HDOS immer, was es gerade gelesen hat und kann sich sowohl vorwärts als auch rückwärts bewegen, in diesem Beispiel von Sektor 10 über 21 bis 22 und zurück.

Ab Byte 13 finden wir den File Information Block:

13	**1F (File type)**
14-23	**Free_space (File Name)**
24-30	**00 00 00 00 00 00 00**
31-36	**00 00 00 00 00 00 00**
37-42	**00 00 00 00 00 00 00**

Mit Byte 13 fängt der 'File Information Block' an, das erste Byte identifiziert den File-Typ, in diesem Fall ist es '1F Hex' (31 dezimal) und das bedeutet ein FREE SPACE file. Jedes File muß durch einen Namen benannt sein, hier ist zwischen Byte 14 und 23 der Name 'Free_Space' geschrieben. Die Bytes 24 bis 30 würden das 'Entstehungsdatum' beinhalten, von 31 bis 36 stünde das Datum, wann dieses File zuletzt gelesen und von 37-42 wann es zum letztenmal geschrieben wurde. Da es sich hier in unserem Beispiel um ein 'Free_Space' handelt, stehen demzufolge noch alle Daten auf Null.
Im Anschluß findet man 16 Bytes 'channel information', die für das 'Free_Space' nicht benutzt werden. Da das 'Free_Space' File kein normales ist, stehen hier natürlich auch keine Informationen, der Rest des 'File information block' steht alos auf Null.
Hinter dem File information block kommt die 'current run adress', was beim 'Free_Space' File die Anzahl der freien Bereiche ist und danach 50 mal 'start adress' und 50 mal 'end adress', was für ein 'Free_Space' File die Start- und Endadressen aller freien Bereiche sind. Bei meiner Disk ist das in diesem Beispiel:

ab Byte 110 folgendes:

```
110-111    05 00 (Anzahl freie
           Bereiche=5)
112-115    4A 01 00 00 (5. Bereich)
116-119    50 00 00 00 (2. Bereich)
120-123    1A 00 00 00 (1. Bereich)
124-127    F8 00 00 00 (3. Bereich)
128-131    33 01 00 00 (4. Bereich)
```

Das sind die 5 Startadressen und ab Byte 312 die 5 Endadressen:

```
312-315    3F 4A 09 00 (5. Bereich)
316-319    AB 00 00 00 (2. Bereich)
320-323    1D 00 00 00 (1. Bereich)
324-327    FF 00 00 00 (3. Bereich)
328-331    38 01 00 00 (4. Bereich)
```

Durch diese Informationen können wir sagen, das die Disk-Nutzung wie folgt aussieht:

```
Sektor   0-     19    Belegt
        1A-     1D    Frei
        1E-     4F    Belegt
        50-     AB    Frei
        AC-     F7    Belegt
        F8-    138    Frei
       139-    149    Belegt
       14A-94A3F      Frei
```

Die Adresse 94A3F hex entspricht 608.831 dezimal, dem letzten Sektor der HDOS area bei: 604 Cylindern x 16 Köpfe x 63 Sektoren.
Die belegten Speicher enthalten natürlich alle unsere Directories und Files und nächstesmal werden wir uns dies alles etwas genauer anschauen. Wie das 'Free_Space' chain auf eurer Festplatte aussieht, könnt ihr ziemlich leicht erfahren, wenn ihr es mit dem Programm 'HDLOOK' anschaut.

Ian D. Spencer, Fichtenweg 10c
53804 Much, Tel. 02245/1657

How to fill a block of memory with a value?

Hi, all SPC members!

Now you'll see some tricks of machine code programming. Ok, for example: you need to fill a block of memory with a value (0 < value < 255). When the block size is 32 bytes then simply write this:

```
LD HL, startadress of block       (40000)
LD DE, startadress of block+1     (40001)
LD BC, size of block -1           (31)
LD (HL), - value                  (180)
LDIR
```

But this one works very slow. This routine goes more faster:

```
LD HL,   40000
LD DE,   40001
LD (HL),180-value you want to fill
         with
LDI      -  1st time
LDI      -  2nd time
LDI      -  3rd time
...
...
LDI      - 31st time
```

Simply you must print LDI 31 times. But what can you do if you need to fill 2 KB or 13 KB? You must use this fastest routine. But it is suitable only for pair length of blocks (2, 4, 100, 2000), but not for 7, 101, 303, 501 etc.:

```
LD (OLD_SP),SP
LD SP,End of block+1
LD H,0-value (i.e. -180)
LD L,H
PUSH HL     if block is 32 bytes you
PUSH HL     must print 16 pushes,
.......     if block is 2048 bytes
.......     then 1024 pushes
.......
PUSH HL
LD SP,(OLD_SP)
RET

OLD_SP DEFW o
```

So, you see that the number of pushes is size/2. Because PUSH works with 2 bytes together.
I think that's all. Hope you'll use this information!

Now some words to other SPC members:

Jean: Why don't you send me TR-DOS disks, it will make our sending life easier!
Bernhard: Keep on TR-DOS!
You: Thanx for reading my articles!

And some advertisment:

If you want to get all the best software from EX-USSR (games, demos, utils - FREE CATALOGUE!) for cheapest prices - send us an e-mail to:

kestlumb@pub.osf.lt
subject: for zhenya





or write us to:

Dalnikovas Eugenijus
Kalvariju g. 142-3
2042 Vilnius/Lithuania

or Sigitas Grigonis
Ateities 1-39
2057 Vilnius/Lithuania

In der Kürze liegt die Würze

oder 1 KB ist genug (Teil 3)

Heute wollen wir uns mal dem Bildschirmrand zuwenden. Bei dem Spiel "Gulpman", besser bekannt als Pacman, fiel mir auf, das nach einem gewonnenen Spiel der Bildschirmrand mit bunten Farben 'geschmückt' war.
Nun, bekanntlich kann man den Bildschirmrand mit dem BORDER-Befehl eine Farbe zuweisen. Er kann aber nur **eine** Farbe annehmen. Was aber, wenn man es gerne etwas 'poppiger' haben möchte? Die Maschinensprache-Programmierer werden jetzt sagen: "Wir schaffen das schon!". Aber - auch die Basic-Programmierer schaffen es. Und das sogar ganz einfach. Gebt das Listing 1 (131 Bytes) ein. Die maximale Farbanzahl ist 10. Man kan auch die in dem Listing gewählten BORDER-Farben ändern, der Fantasie sind keine Grenzen gesetzt.

```
10 REM *** Borderplay ***
20 FOR i=1 TO 100: BORDER 0: BORDER 1:
   BORDER 2: BORDER 3: BORDER 4: BORD
   ER 5: BORDER 6: BORDER 7: PAUSE 1:
   NEXT i
```

Legen wir jetzt noch einen drauf! Das Listing 2 (169 Bytes) erlaubt es, die Randfarben beweglich zu gestalten. Anstatt dem Pausebefehl sind jetzt Doppelpunkte eingefügt. Die Menge hängt immer von der Anzahl der Farben ab. Für 10 Farben wird nur 1 Doppelpunkt benötigt, für neun sind es 13 und für acht 25 Doppelpunkte. Pro Farbe weniger werden also 12 Doppelpunkte mehr gebraucht.

```
10 REM *** Borderplay 2 ***
20 FOR i=1 TO 1000: BORDER 0: BORDER
   1: BORDER 2: BORDER 3: BORDER 4:
   BORDER 5: BORDER 6: BORDER 7::::::
   :::::::::::::::::::: NEXT i
```

Ein besonderes Schmankerl ist Listing 3. Es benötigt gerade 380 Bytes und ist nicht schlecht. Was es macht, verrate ich hier nicht. Probiert es selbst aus; einige von euch werden sicherlich staunen, was der Specci macht.

```
10 REM *** Effektshow ***
20 CLEAR 29999
30 FOR n=30000 TO 30028
40 READ a: POKE n,a: NEXT n
50 DATA 14,255,6,29,33,0,91,62,239,113,
   0,211,254,43,61,194,57,117,5,120,
   194,55,117,13,121,194,50,117,201
60 RANDOMIZE USR 30000
```

Wie immer rufe ich dazu auf, baut die Listings in vorhandene Programme ein, verbessert oder verschönert sie damit und experimentiert herum...

Zum Schluß noch eine Anmerkung zu **"Macht mit - Eine Aktion von Peter Rennefeld"**
Hallo Peter, sende mir doch bitte ein paar Exemplare deiner Aushängezettel zu. Vor ca. 3-4 Jahren führte ich auch eine ähnliche Aktion durch. Ich suchte durch ein Zeitungsinserat insbesondere 128er und Zubehör. Die Folge war, das ich 2 Spectrum 128, einen +2A (wird gerade von Jean Austermühle umgebaut) und ein Opus Laufwerk plus interessante Software, sowie einen Digital Tracer erwerben konnte. Einen dritten 128er Spectrum bekam ich trotz Zusage nicht. Einen ZX80 und ZX 81 suchte ich damals nicht.
Apropos ZX 81! Zwei ZX 81 wurden, man glaubt es kaum, bei der D1-Mission in den Weltraum mitgenommen. Diese hatten die Aufgabe, dafür zu sorgen, daß die Astronauten weder erfroren noch geröstet wurden. Tja, ZX over the world!!!
So, das wärs's für heute...

Rupert Hoffmann, Tulpenstraße 12
92637 Weiden, Tel. 0961/6342321

Der Spectrum kann lesen...

Unter gewissen Bedingungen, die die Verwendung etwas einengen, kann der Spectrum von SCREEN$ Texte erkennen, diese als Bildschirmkopie (nicht COPY) in wählbarem Format ausdrucken (z.B. 6-spaltigen Katalog), als Stringvariable oder als Maschinencode speichern oder weiterbearbeiten, suchen und dergleichen.
Die Bedingung ist, das der SCREEN im Sinclair-Format mit 22 Zeilen zu 32 Buchstaben beschrieben ist. mehr Buchstaben und höhere Zeilenzahl kann er nicht lesen, höchstens gelegentlich einen Buchstaben, der gerade in das Raster paßt.
Dagegen kann er jede Schriftart, auch fremde Alphabete, lesen und interpretiert sie zunächst als lateinische Buchstaben im ASCII-Code oder druckt sie auch mit fremden Download-zeichensatz.
Die Routinen dazu sind einfach und der Druck nur unwesentlich langsamer, dafür wesentlich besser lesbar als die normale COPY-Routine.
Um mich nicht der Gefahr auszusetzen, daß eure SAVE/LOAD und Druckersyntax nicht paßt, gebe ich hier nur das Prinzip an und ihr dürft sie dann selber in ein Programm einbauen.

```
Druck:
    FOR z=0 TO 22: FOR s=0 TO 31:
    LET a$= SCREEN$ (z,s):
    IF a$="*" THEN LET a$="■":
    LPRINT a$;: NEXT s: NEXT z
```

Variable:

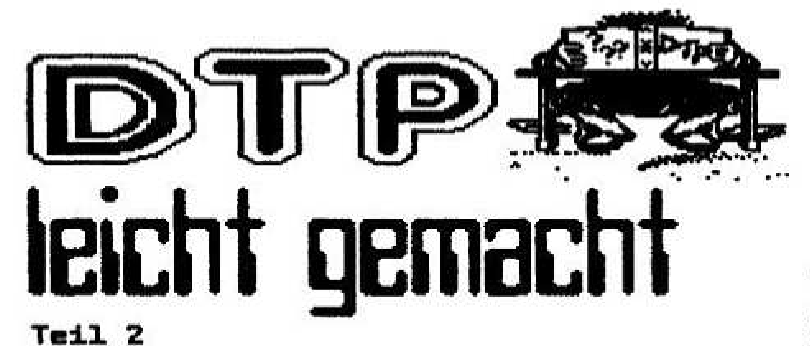
```
LET a$="": FOR z=0 TO 22: FOR s=
0 TO 31: LET a$=a$+SCREEN$ (z,s):
NEXT s: NEXT z
```

Maschinencode:

```
LET a= erste Speicherstelle (z.B.
32000): FOR z=0 TO 22: FOR s=0 TO
31: POKE a, CODE SCREEN$ (z,s):
LET a=a+1: NEXT s: NEXT z
```

Anmerkung: Beim Druck muß das Zeichen "(copyright)" (hier durch ein "•" dargestellt) durch ein anderes ersetzt werden, denn dieses Zeichen versteht der Drucker als: letztes Zeichen löschen und bringt alles durcheinander.

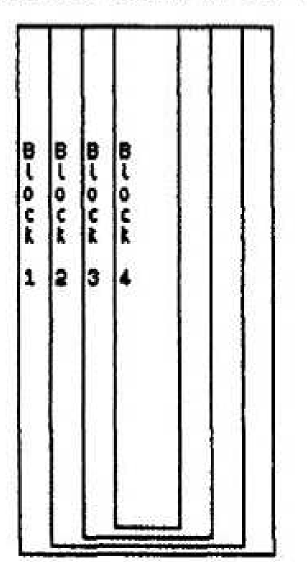
Herbert Hartig, Buchloe

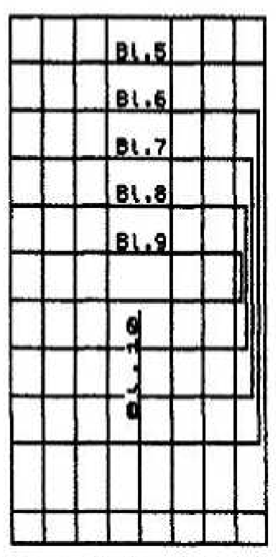


DTP leicht gemacht

Teil 2

Hoffentlich wart ihr letztes Mal etwas schlauer als ich, denn ich hatte vergessen euch mitzuteilen, wie man sein mühsam erstelltes Werk abspeichert. Dies geschieht auch wieder aus dem Hauptmenu heraus über die Taste "S" und der Angabe eines Filenamens, wobei ihr mit "D" wieder auf euer Speichermedium zurückgreift. Eigentlich denke ich, das dies auch wirklich keinem von euch Mühe bereitet hat, das Problem war aber wahrscheinlich für manch einen ein ganz anderes.
Wir befanden uns zuletzt nämlich im Typeliner-Menu. Und um in das Hauptmenu zu gelangen, muß man dieses erst einmal verlassen. Und das kann man nur, wenn man auf der Seite ist, wo auch das "DIN A4 Blatt" abgebildet wird. Über die Taste "Inverse Video" gelangen wir nun zurück ins Hauptmenu.
Nun aber zurück zur Aufgabenstellung aus dem lezten Info. Ich weiß zwar recht wenig darüber, wie ihr zurechtgekommen seid, aber wenigstens eine Lösung wurde mir von Herrn Hartig zugeschickt.
Nun - in diesem Fall gab es, ähnlich wie beim Programmieren, verschiedene Lösungen. Man konnte z.B. die gesamten Felder mit einzelnen Linien (Stylemodus 1 und 2) nachzeichnen. Nicht falsch, aber: wir hätten glatte 21 Linienblöcke verbraucht (von 24, die zur Verfügung stehen), und zwar 12 senkrechte und 9 waagerechte. Wer auf diese Art später einmal versucht Tabellen zu erstellen, wird sehr oft böse Überraschungen mit den Linienanschlüssen erleben. Denn Achtung: Beim Spectrum gilt nicht unbedingt das "Wysiwyg" (What you see is what you get), zumindest nicht, was einzelne Linien anbetrifft.
Nun, eleganter und blocksparender ging es sicher über Rahmen im Stylemodus 3.
Meine Lösungsvorschlag benötigt genau 10 Linienblöcke und sieht in der Entstehung so aus:





4 Blöcke (Style 3) decken bis auf den Mittelstrich alle senkrechten Linien ab. Zur Verdeutlichung habe ich diese unterschiedlich hoch gehalten. Dasselbe geschieht nun mit den 5 waagerechten Blöcken (auch hier zur Verdeutlichung etwas kürzer gehalten). Block 10 (Style 1) letztendlich ist die alles komplettierende Mittellinie.
Herr Hartig schrieb mir, seine Lösung bestünde aus 9 Blöcken. Wenn er sich hier nicht vertan hat, dann bin ich auf seine Lösung sehr gespannt. Ich habe es - ganz ehrlich - nicht mit 9 einzelnen Blöcken geschafft.
Habt ihr mal die Abstände zwischen den Linien gemessen? Das sollte bei euch so aussehen:

2,54	5,1 2,4	7,6	10,1	12,7	15,2	17,7	20,2
		4,9					
			7,45				
				10,0			
					12,5		
						15,1	
							17,6
							20,2
							22,7
DTP-Raster STAR LC 10							25,2
							26,9

Waagerecht erhalten wir immer das Vielfache eines Zoll (2.54 cm). Das oben gezeigte Schema kann uns jetzt also helfen, in etwa ein Format festzulegen, was wir im nächsten Teil gleich nutzen werden.

Wo von WoMo

BASIC für Anfänger

TEIL 6

Hallo Freunde !

Wieder einmal geht es darum dem Basic unserer Rechner näherzukommen.
Heute beginnen wir mal mit dem Startbefehl für die Programme. RUN - Damit wird normalerweise ein Programm bei Zeile 1 gestartet. Allerdings werden dabei alle Daten und Variablen gelöscht. Wenn die Variablen erhalten bleiben sollen, dann nehmt besser GOTO 0, damit beginnt das Programm zwar ganz am Anfang, aber die Variablen bleiben erhalten, bis sie durch das Programm selbst neu gesetzt werden.
Also achtet bei Programmabbrüchen auf die Zeilennummer in der Meldung. Oft kann das Programm nach einem Bedienungsfehler mit dieser, oder der nächstfolgenden Zeilennummer wieder gestartet werden, ohne dabei wertvolle Daten zu verlieren. Allerdings müßt ihr dann mit GOTO XXX starten.
Wenn ihr beim Programmieren einen Probelauf machen wollt, und etwa die Anzahl von Durchläufen einer Schleife wissen wollt, dann programmiert doch einfach eine Additionsschleife mit hinein:

```
1000 LET a=a+1
```
(a natürlich am Programmanfang definiert!)

Wenn ihr dann den Durchlauf beendet habt, oder abgebrochen wurde, braucht ihr nur noch PRINT a einzugeben. Nach erfolgter Programierung kann die Zeile ja wieder gelöscht werden.
Jetzt wollt ihr ein Programm so abspeichern, daß es nach dem erneuten Laden sofort startet. Kein Problem:

```
9998 STOP
9999 SAVE "Test" LINE 200
```

Diese Zeilen bewirken, daß bei Probeläufen vor dem Savebefehl gestopt wird, und das LINE 200 sagt, daß nach erneutem Laden bei Zeile 200 gestartet wird (Ohne Angabe einer Zahl bewirkt LINE den Start bei Zeile 0).
Microdrivebesitzer können ein Programm pro Cartrige zum Schnellstart vorbereiten, indem sie ihm den Namen "RUN" geben. Nach dem Einschalten brauchen sie nur RUN einzugeben und das Programm wird automatisch geladen.
Beim Plus D heißt der Titel "Autoload" (Das funktioniert nur, wenn auf dieser Disk auch ein "+sys" File ist. Daher statte ich jede Diskette nach dem formatieren mit einem "+sys" File aus.) Leider geht es dabei nur nach dem Einschalten, da ein Reset das Plus D nicht beeinflusst.

Jetzt nochmal zu den Befehlen PEEK und POKE.

Wenn ihr euch ein Programm schreibt, um jedes Byte des ROM's mal auszulesen und in verschiedenen Formen (Dezimal, hexadezimal und auch als Binärmuster) darzustellen, werdet ihr am Schluß des ROM's den Zeichensatz in Binärer Form finden. Das zeigt, daß der Computer das Muster seiner Schrift erstmal irgendwo nachlesen muß.
Jetzt muß der Rechner ja irgendwie erfahren, wo dieser Zeichensatz steht. Also gibt es eine Adresse im veränderlichen Teil des Speichers, die den Rechner instruiert, wo der Zeichensatz ist. Und genau da können wir sinnvoll eingreifen. Erstens können wir jedes gewünschte Binärmuster in dezimaler Form eingeben, und zweitens können wir die Adresse, wo der Rechner sich seine Zeichen suchen soll, neu eingeben.
Auf diese Weise ist es möglich, dem Rechner einen völlig neuen Zeichensatz vorzugeben.
Den neuen Zeichensatz einzugeben erfordert viel Arbeit, da ja jede Zeile einzeln eingepoket werden muß. Es lohnt sich, diese Zeichen erst auf einem Blatt zu zeichnen, dann die entsprechenden Binärzahlen aufzuschreiben und dann erst Stück für Stück einzugeben. Am Schluß gibt es ein Beispiel zu der Binärumsetzung.
Wenn der neue Code komplett eingegeben ist, braucht ihr nur noch die Adresse zu ändern, und alles, was ihr programmiert oder sonstwie auslesen wollt, wird im neuen Zeichensatz dargestellt. Die Adresse gebt ihr wie folgt ein:

```
POKE 23606,XXX:POKE 23607,YYY
```

Wenn ihr die Zahlen 0 und 60 nehmt, wird der alte Zeichensatz eingeblendet. Wenn ihr die Zahlen 0 und 249 verwendet, wird ein neuer Zeichensatz verwendet, der bei 64000 beginnt (Da die ersten 32 Zeichen Steuerzeichen sind, beginnt der "echte" Zeichensatz erst um 256 (32*8) Byte höher, was zur Folge hat, daß die Adresse, die ihr angeben müsst, genau 256 tiefer ist. In diesem Fall 63744 - für die Variablen

umgerechnet ergibt eben 249 in der zweiten Stelle.).
Auf diese Weise ist es etwa möglich, in einem Programm mehr als zwanzig Sonderzeichen zu definieren, indem ihr jeweils den erforderlichen Zeichensatz für die nächste Ausgabe vorher anwählt.
Um den neuen Zeichensatz abzuspeichern verwendet ihr den Befehl

```
SAVE "Zeichen1" CODE 64000,XXX
```

Mit diesem Befehl werden alle Daten ab der Speicheradresse 64000 bis zu Adresse 64000 +XXX so wie sie kommen, abgespeichert. Einlesen könnt ihr diese Daten mit:

```
LOAD "Zeichen1" CODE
```

Beim Laden erkennt der Computer selbstständig wohin ihr wieviel Daten schicken wollt. Wenn ihr die Länge wißt, oder beeinflussen wollt, und eine neue Adresse nötig ist, etwa weil der Platz ganz oben schon belegt ist, dann könnt ihr auch

```
LOAD "Zeichen1" CODE XXXXX,YYYYY
```

eingeben, das lädt die Daten dann auf eine neue Adresse und mit der angegebenen Länge. Auf diese Weise könnt ihr Daten in den verschiedensten Programmen verwenden, ohne sie jeweils neu eingeben zu müssen. Damit können nicht nur Zeichensätze eingelesen werden, sondern z.B. Bilder, die als SCREEN$ abgespeichert sind, können ganz woanders hingeladen werden.
SCREEN$ bedeutet nämlich nichts anderes als CODE 16384,6912. Ladet mal ein Titelbild mit

```
LOAD "Test" CODE 40000,6912
```

dann könnt ihr dieses Bild mit dem MC-Programm des letzten Beitrages einblenden oder sonstwie verschieben.
Mit dem Befehl MERGE, der ja ein dazuladen von Daten zu bereits existierenden Daten erlaubt, könnt ihr auch mehrere Teile miteinander verbinden.
So - jetzt noch zu den Befehlen GOSUB, GOTO und RETURN. Die erste Auswirkung ist bei GOTO und GOSUB gleich. Es wird zu der angegebenen Zeilennummer gesprungen. Allerdings könnt ihr an die Stelle, von wo aus ihr gesprungen seid, nicht so einfach zurück. Gebt ihr nach dem Unterprogramm einfach eine weitere GOTO Zeile ein, so könnt ihr nur jeweils genau an diese eine Stelle zurückspringen.
Oft soll aber ein Unterprogramm von verschiedenen Stellen aus angesprungen werden, Daten verarbeiten, und dann soll an der vorherigen Stelle weitergemacht werden. Auf diese Weise kann es mehrfach genutzt werden, und Speicherplatz sparen.
Jetzt gibt es die Möglichkeit, vor jedem Sprung eine Variable zu setzen, in der die entsprechende Zeile steht, und am Ende des Unterprogramms diese Variable hinter den GOTO Befehl zu setzen. Aber dafür gibt es einen Befehl, der genau das erledigt. GOSUB setzt, ohne daß ihr es merkt eine Variable im Speicher, in der die Zeilennummer für den Rücksprung notiert wird (Um eine Zeilennummer höher, als die Zeile, die den GOSUB Befehl enthält).
Steht dann im Unterprogramm RETURN, so springt der Rechner genau zu der Zeile zurück, die der vorher ausgeführten GOSUB Zeile folgt.

Zu den FOR NEXT Schleifen gibt es noch etwas zu schreiben: Wenn ihr die Zahlenwerte der Variablen braucht, aber die normale Abfolge nicht euren Wünschen entspricht, dann nehmt den STEP-Befehl.

```
10 FOR a=1TO 20 STEP 4
20 NEXT a
```

Diese Schleife wird nur fünfmal durchlaufen, weil jedesmal vier dazugezählt werden. Es geht aber noch besser:

```
10 FOR a=-20 TO -40 STEP -2.6
20 NEXT a
```

Bei dieser Schleife werden vom negativen Ausgangswert jeweils noch 2.6 abgezogen, bis a gleich oder kleiner (negativer) als -40 ist. In diesem Falle ist es -40.8
Es können also positive und negative Zahlen für die Schleife verwendet werden, als auch unabhängig davon positive und negative Zahlen für den Schrittwert. Allerdings müßt ihr darauf achten, keinen Unsinn zu programmieren.

```
10 FOR a=10 TO 20 STEP -1
```

Diese Zeile ist solch ein Unsinn. Der Rechner würde hierbei den Inhalt der Schleife glatt überspringen. Wobei

```
10 FOR a=20 TO 1 STEP -2
20 PRINT a
30 NEXT a
```

zu dem Resultat führen würde, jeweils eine kleinere Zahl anzuzeigen, bis die Schleife durchlaufen ist.
Auch solltet ihr aufpassen, nicht dieselbe Variable, die die Schleife steuert, innerhalb der Schleife zu verwenden - das kann euer Ergebnis verändern (außer natürlich, ihr braucht genau diese Veränderung).

```
10 FOR a=1 TO 20
20 LET a=a-1
30 NEXT a
```

Solltet ihr so etwas eingeben, so bringt bitte Geduld mit, der Rechner ist dann nämlich ziemlich lange beschäftigt (genau gesagt, bis zum nächsten Stromausfall).

Der letzte Befehl für heute ist PAUSE.

Genau wie einige von euch vermutet haben, dient dieser Befehl dazu, dem Rechner mal eine kleine Erholung vor allzu lästigen Programierern zu gewähren. Allerdings müßt ihr ihm sagen, wie lange er vor euch verschont bleibt:

```
10 PAUSE 250
```

würde den Rechner in Europa für fünf Sekunden anhalten. Da in Amerika alles anders ist, auch die Netzfrequenz, und der PAUSE Befehl sich danach richtet, so braucht ihr dort für dieselbe Zeit die Zahl 300.
Wollt ihr den Rechner für unbestimmte Zeit anhalten, so gebt nur eine Zahl größer 65535 ein. Günstig ist 4E4 (4-Exponent-4), weil das englisch ausgesprochen wie 'for ever' klingt (Dadurch wird das im Listing deutlicher). Es funktioniert aber auch mit PAUSE 0.
Bei jedem PAUSE Befehl wird nur so lange angehalten, bis der Rechner von irgendwem befummelt wird. Wenn also jemand eine Taste drückt, so macht er sofort weiter.
So - und jetzt gebt mal folgendes ein:

```
10 FOR a=0 TO 255
20 POKE 64256+a,a
30 NEXT a
40 POKE 23607,249
50 LIST
```

Die Adresse habe ich hierbei extra um 256 nach oben gerückt, damit die Großbuchstaben und nicht die Zeichen verändert werden, um das Ergebniss einer solchen Änderung zu zeigen.
Wenn euch Tasword und seine Nachfolger bekannt ist, so kennt ihr bereits einige der sinnvollen Anwendungen, die damit möglich werden.

Und zum guten Schluß noch ein paar nützliche POKE's, welche ihr in euren Programmen oder beim programmieren gut brauchen könnt:

POKE	Standard-wert	Wirkung:
23561	35	Zeit in 1/50 sec. bis Beginn der Tastenwiederholung. Je kleiner der Wert, umso schneller beginnt die Wiederholung.
23562	5	Zeit zwischen 2 Wiederholungen.
23606 -23607	0 60	Adresse des aktuellen Zeichensatzes abzüglich 256 (vom Gesamtwert)
23609	0	Länge des Keybordklicks. Kann zwecks akkustischer Tastaturbestätigung gut geändert werden.
23624		Farbe des Border mal 8. Kann zwecks grafischer Gestaltung verändert werden
23659	2	Zeilenzahl für Fehlermeldungen etc. Wenn 0, dann keine Fehlermeldungen möglich (Listschutz)
23675 -23676	88 255	Adresse des ersten selbstdefinierten Grafiksymbols. Kann bei Speichermangel geändert werden.
23693	56	Aktuelle Farben. Kann geändert werden, um PAPER und INK zu ändern.
23730 -23731		RAMTOP-Adresse des letzten für Basic verwendbaren Bytes. Kann sinnvoll geändert werden. (Siehe BASIC 5)
23791	0	Gibt Zahl der Kopien für MICRODRIVE an. Je höher, desto schneller der Zugriff auf ein File (sinnvoll bei kurzen, aber wichtigen Programmen z.B.: Kopierprogrammen)

Die POKE's und einige andere Daten sind dem Buch "ZX Spectrum Tips und Tricks" von Data Becker entnommen. Einiges davon habe ich selbst sehr erfolgreich benutzt (Als ich noch mit Kassette gearbeitet habe, hatte ich Copyplus mit der Zahl 100 in POKE 23791 auf MICRODRIVE - sicherer und extrem schneller Zugriff war das Ergebnis).
Es gibt noch viel mehr interessante Adressen - lest mal die Infos genau durch, immer wieder wird mal die eine oder andere genannt. Außerdem kann man POKE's mit entsprechenden Geräten (z.B.: Multiface) in Spiele eingeben, was dann neue Möglichkeiten gibt.
Nach meiner Meinung muß man nicht genau wissen, was genau der eine oder andere POKE bewirkt, wenn er die gewünschte Wirkung innerhalb des Programmes erfüllt.
Nur sollte man ein neues Programm bei der Verwendung von POKE's gut und genau auf eventuelle Fehlfunkionen durchtesten, um vor unliebsamen Überraschungen gefeit zu sein.
Probiert die obengenannten POKE's mal ein bißchen aus, im Notfall könnt ihr immer noch den Stecker ziehen.

Viel Spaß wünscht euch bis nächsten Monat

Peter Rennefeld, Genhodder 19
41179 Mönchengladbach, Tel. 02161/571141

Adventure Quest

Hallo Adventure Freunde!!

Diesesmal wollen wir uns mit dem Programm **"Adventure Quest"** beschäftigen, einem weiteren Vertreter aus der Jewels of Darkness Trilogie. Die Anzahl der Locations ist bei solch komplexen Programmen erfahrungsgemäß sehr groß was oft sehr leicht zur Desorientierung führt. Wir haben deshalb bei dieser Lösung nur solche Räume in den Plan mit aufgenommen, die wir auf unserem Lösungsweg auch passieren müssen. Dies ist also wahrscheinlich keine komplette Darstellung aller möglichen Locations von Adventure Quest, doch ist sie für unsere Belange ausreichend. Wer will, kann den Plan ja weiter ausarbeiten.
Kommen wir nun zu den einzelnen Locations des beiliegenden Planes:

001) At the end of a road
002) inside the building / bottle, bunch of keys, fruit, sling, table
003) In a narrow north-south valley
004) In a north-south valley
005) In a 20 foot depression
006) in a steeply sloping gully / orchid
007) In a woodland above the valley
008) Path through dense forest
009) In a grove of tall trees
010) In a small woodland clearing / pan pipes, ancient medallion
011) In a dense forest
012) Lost in the forest A
013) Lost in the forest B
014) Lost in the forest C
015) Lost in the forest beside a huge oak tree
016) Sitting on a gnarled treebranch / silverball
017) On a steep eastside of a valley / onion
018) The base of a stone pinnacle
019) Path up the rock pinnacle
020) At the top of the pinnacle of Obdurat / stick
021) On a road north of the building
022) At the edge of the fertile land
023) Southern edge of a vaste desert
024) On a trackless desert A
025) On a trackless desert B
026) On a trackless desert C
027) On a trackless desert D
028) On a trackless desert E
029) On a trackless desert F
030) On a rocky outcrop
031) In a sheltered wadi
032) At wadi end
033) On the side of a mountain
034) On a stone staircase A
035) On a stone staircase B
036) On a wide stone staircase
037) On an east-west track
038) On a climbing path
039) In a sheltered east-west valley
040) In a dry canyon
041) In a dry east-west canyon
042) In a looted treasure cave / sundial
043) Top of the pyramid
044) Inside a small temple
045) Track up the mountains
046) On a track leading upwards
047) On a track leading backround the mountain
048) On stone steps
049) On steps leading upwards / giant rocks
050) On a bleak rocky mountain side
051) On a steep path through sparse vegetation
052) Standing on the snowfield / snowman
053) Outside the guard tower
054) North tower room
055) South tower room
056) On a tight spiral staircase A
057) On a tight spiral staircase B
058) Outside a small door
059) Small gloomy room
060) In a tiny alcove / emerald eye, leather bag
061) Small guard room / rope
062) Outside the south door of guard tower
063) On a stone staircase C
064) North edge of an oasis / Djiin
065) In an oasis / lamp
066) Swimming in a pool of water / trident
067) In a smelly cave / stalagmite
068) On a ledge A
069) On a ledge B
070) On a ledge above rapids
071) In a quiet pool below rapids
072) On a small gravel beach / net
073) Beside a huge clam
074) On a featureless lakebed
075) In a drowned graveyard
076) On the lakebed above a trench A
077) In shallow water at the edge of the lake
078) South of a huge door
079) On the lakebed above a trench B / shark
080) On the lake bed
081) Inside the drowned church / jelly fish

Adventure Quest

mapped by

Double 'H' Software

082] At the westend of the underwater trench
083] At the eastend of the underwater trench
084] South of a magnificence cave
085] Steeply sloping east-west tunnel
086] Junction between north-south and east-west passages
087] On a ledge clinging to the west wall
088] Center of magnificent cave
089] North of a magnificent cave
090] In a small dead-end room / statue
091] Clinging to a web covered stalagmite
092] Middle of a huge web
093] In the lair of a giant spider / earth-stone
094] On a ledge which clings to the west wall of a large cavern
095] Entrance to an orc lair
096] Jagged north-south passage
097] At the end of a passage
098] In a tiny store room
099] In an east-west passage
100] In a sloping corridor A
101] In a sloping corridor B
102] In a round smooth cave / dragon
103] In the dragon's lair / egg
104] In a cave overlooking a crater
105] In a ledge over the crater
106] On a ledge north-east of the crater
107] Ledge north-west of the crater
108] Ledge west of the crater
109] Ledge south-west of the crater / cloak
110] Ledge south-east of the crater
111] Ledge east of the crater
112] Cave opening onto a crater
113] On a spiral ramp A
114] On a spiral ramp B
115] On a spiral ramp C
116] Standing on shattered rocks
117] In a warm passage
118] On red-hot coals
119] In an ornate room
120] Hot east-west corridor
121] East side of a bottomless chasm
122] On a narrow stone bridge
123] On the west of the chasm
124] Standing by the altar / sun-stone
125] On a bleak moor A
126] On a bleak moor B
127] On a bleak moor C
128] On a bleak moor D
129] On a hilltop / star-stone, brazier
130] Tiny ledge
131] On a bleak moor E
132] On a bleak moor F
133] On a bleak moor G
134] On a bleak moor H



Der 1. Schritt führt euch hierher...

135] On a bleak moor I
136] On a steep path
137] On steps leading behind a waterfall
138] On steps behind a waterfall
139] On slippery steps
140] At the foot of a flight of steps
141] On a north-south path
142] Circle of silent stones
143] On a narrow east-west path
144] Lost in the marsh A
145] On a north-south path
146] Lost in the marsh B
147] On a path deep in the marsh
148] On an east-west causeway
149] On an island outside an ancient house
150] In an ancient panelled entrance hall
151] In the main hall
152] In a high room / boots
153] On a small ledge
154] In deadly quicksand / ancient medallion
155] On a dark granite ramp
156] On a wide ramp up the tower
157] Below the door of rock
158] Below the door fo gold
159] Below the door of silver
160] Below the door of glass
161] At the south of a magnificent throne-room
162] crossing of passages
163] In an east-west corridor
164] In an east-west passage
165] At a crossover between passages
166] Surrounded by doorways
167] Hidden in a curtained alcove
168] At an open doorway
169] In a long dusty north-south passage
170] Top of a stairway
171] Gloomy stairs leading downwards
172] At the south of a pit
173] Beside a Bane-fire

Soweit die Locations zum beiliegenden Plan. Damit habt ihr schon mal die Möglichkeit, Euch etwas mit den Gegebenheiten vertraut zu machen. Nächsten Monat gibt es dann die dazugehörende Lösung. Bis dahin viel Spaß!

Harald R. Lack, Heidenauer Str. 5, 83064 Raubling
Hubert Kracher, Starenweg 14, 83064 Raubling

Deutsche Übersetzung des TECHINFO.DOC des Sinclair ZX Spectrum Emulator 'Z80' v3.05 - 11/11/96 - von G.A. Lunter

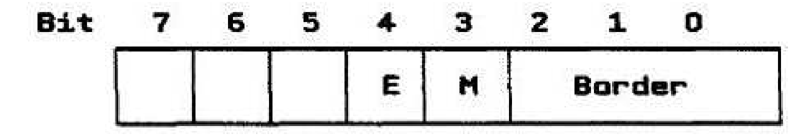
Vorwort zu dieser Übersetzung: Ich habe - soweit mein Englisch reicht - diesen, meiner Meinung nach sehr interessanten Text ins Deutsche übersetzt. Für Fehler, Fehl-Interpretationen oder ähnliches kann ich keine Gewähr übernehmen, würde mich aber freuen, wenn mir in einem solchen Fall jemand Bescheid geben könnte! Danke, Bernhard.

5. TECHNISCHE INFORMATIONEN

Inhalt:

5.1 Der Spectrum 48K

In diesem Abschnitt wird die Hardware des 48K Spectrum diskutiert. In diesem Abschnitt, bezieht sich 'Spectrum' immer nur auf das 48K Modell.
Der Spectrum ist auf der Hardware Basis eine sehr einfache Maschine. Es gibt das 16K ROM welches den untersten Teil der Speicher-Adressen belegt und 48K RAM welche den Rest belegen. Ein ULA (Uncomitted Logic Array) welches die untersten 6912 Bytes des RAM als Bildschirmspeicher benutzt, und lediglich die Logik für nur einen I/O Port komplettieren die Maschine, aus Sicht der Software, schließlich.
Jede gerade I/O Addresse spricht den ULA an, doch um Probleme mit anderen I/O Hardware-Teilen zu verhindern, sollte nur Port FE benutzt werden. Wenn der Port geschrieben wird haben die Bits folgende Bedeutung:



Bit 7	6	5	4	3	2	1	0
			E	M	Border		

Die untersten drei Bits wählen die Border- (=Rand) Farbe; eine 0 in Bit 3 activiert den MIC Ausgang, und eine 1 in Bit 4 activivert den EAR Ausgang (welcher den internen Lautsprecher ansteuert). Der reale Spectrum activiert ebenso den MIC wenn auf den EAR-Ausgang geschrieben wird; der Emulator macht das nicht. Das ist kein Problem; der MIC Ausgang wird nur für das Speichern verwendet, und wenn gespeichert wird, aktiviert der Spectrum den internen Lausprecher nie. Die obersten drei Bits werden nicht benutzt.
Wenn von Port FE gelesen wird, sind die obersten 8 Adresse-Linien auch wichtig. Eine 0 an einer dieser Linien wählt eine bestimmte Halb-Reihe von 5 Tasten:
IN: Liest Tasten (Bit 0 bis Bit 4 inclusive, in dieser Reihenfolge)

#FEFE	SHIFT, Z, X, C, V
#EFFE	0, 9, 8, 7, 6
#FDFE	A, S, D, F, G
#DFFE	P, O, I, U, Y
#FBFE	Q, W, E, R, T
#BFFE	ENTER, L, K, J, H
#F7FE	1, 2, 3, 4, 5
#7FFE	SPACE, SYM SHFT, M, N, B

Eine 0 in einem der untersten 5 Bits bedeutet das die zugehörige Taste gedrückt ist. Wenn mehr als eine Adressen Zeile auf 0 gesetzt wird, wird das Resultat ein logisches UND aller einzelnen Eingaben sein, das bedeutet daß zumindestens eine der zugehörigen Tasten gedrückt wird. Zum Beispiel: Nur wenn jeder der untersten 5 Bits des Lese-Resultats von Port 00FE (zum Beispiel durch XOR A/IN A,(FE)) eine 1 ist, ist keine Taste gedrückt.
Eine abschließende Bemerkung zur Tastatur. Sie ist in einer matrixmäßigen Form verdrahtet, mit 8 Zeilen und 5 Reihen, wie aus den obigen Bemerkungen zu ersehen ist. Jede zwei Tasten, die zusammen gedrückt werden, können durch Lesen der IN Ports separat decodiert werden; aber auch, wenn mehr als zwei Tasten zusammen gedrückt werden, können diese evtl. nicht genau decodiert werden. Als Beispiel:
wenn man Caps shift, B und V zusammen drückt, wird der Spectrum auch denken, das die Space Taste gedrückt ist und reagiert mit der Fehlermeldung 'Break into Program'. Dieses Matrix-Verhalten wird auch emuliert, ohne es würde z.B. Zynaps nicht pausieren, wenn man die Tasten 5,6,7,8 und 0 gleichzeitig drückt.
Bit 6 (Wert 64) des IN-Port FE ist das EAR Eingang Bit. Wenn es auf diesem Eingang ruhig ist, ist der Wert 0, außer beim frühen Modell 2 des Spectrum, wo es 1 ist. Wenn ein Signal anliegt, wechselt dieses Bit entsprechend. Die Spectrum Lade-Software ist nicht empfindlich zur Polarität dieses Bits (was auch definitiv nicht erwünscht ist, bedingt durch die unterschiedlichen Modelle), und auch weil niemand wissen kann, ob der Kassetten-Recorder nicht die Signal Polarität umdreht!). Einige ältere Programme beziehen sich auf das Faktum, das Bit 6 immer 1 ist (z.B. Spinads); für diese Programme kann der Emulator einen Modell 2 Spectrum darstellen.

Bits 5 und 7 sind immer 1 (außer in einigen Clones; Einar Gattoni Saukas sagte mir das der TK-90X das Bit 7 auf 0 setzt).
Die ULA mit den unteren 16K des RAM, und der Prozessor mit den oberen 32K RAM und 16K ROM arbeiten unabhängig voneinander. Die Daten- und Adress-Busse des Z80 und die ULA sind durch kleine Widerstände verbunden und normalerweise trennt dies effektiv die Busse. Wenn aber der Z80 in die unteren 16k RAM schreiben oder von dort lesen und die ULA auch dorthin zugreifen will stoppt die ULA den Prozessor und nachdem sie fertig gelesen hat bekommt der Prozessor wieder Zugriff auf den unteren Speicher über die Widerstände. Ein sehr schnelles, billiges und nettes Design - in der Tat!
Wenn man ein Programm in den unteren 16K des RAM ablaufen läßt bzw. schreibt oder liest in diesem Speicher-Bereich, wird der Prozessor somit manchmal angehalten. Dieser Teil des Speicher ist somit etwas langsamer als der obere 32K Block. Das ist auch der Grund warum man keine Sound - oder Save-Routine im unteren Speicher-Bereich schreiben kann; das Timing wäre nicht exact, und die Musik wird sich schlimm anhören. Desweiteren, bringt ein IN vom Port FE den Prozessor zum anhalten, weil die ULA das Resultat liefern muß. Deswegen ist das Lesen vom Port FE im Durchschnitt ein wenig langsamer als das Lesen von anderen Ports; normalerweise dauert eine IN A,(nn) Instruktion 11 T States, aber im Durchschnitt 12.15 T States wenn nn=FE ist. Siehe weiter unten für mehr exacte Informationen.
Wenn der Prozessor von einem nicht-existierenden IN Port liest, z.B. FF, wird die ULA nicht gestoppt, und es wird auch nichts auf den Daten-Bus gelegt. Darum wird man eine Mixtur aus FF's (=besetzter "idle" Bus), Screen und ATTR Daten Bytes (nebenbei bemerkt: das letztere sehr knapp) erhalten. Das wird nur passieren wenn die ULA den Bildschirmspeicher liest, 61.5% (=Verhältnis von 192/312) einer 1/50tel Sekunde wenn ein Bild dargestellt wird. Die anderen 38.5% der Zeit erstellt die ULA den Border oder generiert einen vertikalen Rücklauf. Dieses Verhalten wird von einigen Programmen benutzt, z.B. von by Arkanoid, und Z80 emuliert auch dies.
Abschließend gibt es noch einen interessanten Fehler in der ULA, welcher auch mit dem geteilten Bus zu tun hat. Nach jedem Instruktionen Fetch Zyklus des Prozessors legt dieser das I-R Register-"Paar" (nicht das 8 Bit interne 8 Bit Instruktionen Register, sondern das Interrupt und das R Register) auf den Adress-Bus. Die unteren 7 Bits, das R Register, wird für den Speicher Refresh benutzt. Wie auch immer, die ULA gerät durcheinander, wenn I einen Wert von 64-127 hat, weil sie meint der Prozessor will von den unteren 16K RAM sehr, sehr oft lesen. Die ULA kommt mit dieser Lese-Frequenz nicht mehr mit und verliert regelmäßig ein Screen Byte. Anstatt des aktuellen Bytes wird das vorher gelesene Byte dazu verwendet, um das Video-Signal zu erzeugen. Der Bildschirm sieht aus, wie wenn er mit "Schnee" gefüllt wäre, der Spectrum stürzt aber nicht ab, und das Programm läuft auch normal weiter. Es gibt ein Programm, das ich kenne, und das dieses benutzt um einen netten Effekt zu erzeugen: Vectron. (Welches, nebenbei gesagt, sehr nette Musik hat). Dieser Effekt ist nicht implementiert, weil er eigentlich auch ziemlich nutzlos ist (doch vielleicht werde ich auch dies in einer zukünftigen Version implementieren).
Der Prozessor hat drei Interrupt-Modi, ausgewählt durch die Instruktionen IM 0, IM 1 und IM 2. Im Modus 1 fährt der Prozessor einfach eine RST #38 Anweisung aus, wenn ein Interrupt auftritt. Das ist der Modus in dem der Spectrum normalerweise läuft.
Der andere Modus welcher gewöhnlich benutzt wird ist IM 2. Wenn ein Interrupt auftritt bildet der Prozessor zuerst eine 16 Bit Adresse indem er das I Register (als High Byte) zusammen mit dem kombiniert was auch immer das unterbrechende Gerät auf den Daten Bus legt. Nun liest der Prozessor die 16-Bit Adresse von diesem Interrupt Tabellen Eintrag und abschließend führt er einen CALL zu der Unter-Routine an dieser Adresse aus.
Rodnay Zaks in seinem Buch 'Programmierung des Z80' erklärt, das nur gerade Bytes als niederwertiges Index-Byte erlaubt sind, aber das ist nicht wahr. Der normale Spectrum enthält keine Hardware um ein Byte auf den Bus zu legen, und der Bus als solcher hat immer den Wert FF (weil auch die ULA den Screen nicht liest wenn sie einen Interrupt generiert), somit ist die resultierende Index-Adresse 256•I+0FF. Wie auch immer, manche nicht-so-nette Hardware-Geräte legen Dinge auf den Daten-Bus wenn sie es nicht sollten, so das spätere Programme es nicht mitbekommen, daß das niedrige Index-Byte 0FF ist.
Diese Programme enthalten eine 257 Byte Tabelle von gleichen Bytes die ab 256•I startet, und die Interrupt-Routine ist an einer Adresse, die ein Vielfaches von 257 ist. Ein nützlicher aber nicht so gebräuchlicher Trick ist es, die Tabelle nur FF's enthalten zu lassen (oder das ROM dafür zu benutzen) und ein Byte 18 hex, den Opcode für JR, an Adresse FFFF zu legen. Das erste Byte des ROM ist ein DI, F3 hex, so daß der JR zu FFF4 springt, wohin man einen langen Sprung mittels JP legt, um die aktuelle Interrupt-Routine anzuspringen.
(Fortsetzung im nächsten Info)

Bernhard LUTZ, Hammerstr. 12, 76756 Bellheim
Tel. 07272-77372 (b. Sprenger, Mo-Do ab 18 Uhr)
Fax/AB/Mailbox: 07272-92108
email: luzie@t-online.de



Warum den +2A/B oder +3 umbauen ???

Ich habe vor kurzem festgestellt, daß selbst Besitzer von diesen Rechnern nicht unbedingt wissen, welchen Sinn der Umbau, von dem immer mal wieder geschrieben wird, hat.
Daher will ich mal ganz laienhaft die Gründe nennen:
Zum ersten werden bei dem normalen Umbau die Joystickports auf die alte Sinclair-Norm umgerüstet, da die neueren Rechner eine eigene Norm haben, bei der sich nur noch speziele Joysticks anschließen lassen. Nach erfolgtem Umbau können die Joysticks der alten Atari-Norm (wie etwa Quickshot II) wieder angeschlossen werden (gilt übrigens auch für den grauen +2).
Dann wurde der Expansionsport geändert, was zur Folge hatte, daß alle alten Interfaces nur noch über einen Adapter, den sogennanten Fixit betrieben werden können - und wer hat den schon, außer den Plus D Besitzern?
Nach dem Umbau kann auf diese Störquelle (je mehr Stecker, desto mehr Möglichkeiten für Wackelkontakte) verzichtet werden. Mir ist kein Interface bekannt, welches die neue Norm braucht.
Dann gibt es im ROM Unterschiede, die mir zwar nicht genau bekannt sind, auf die aber in einigen Demos ausdrücklich hingewiesen wird. Beim Umbau wird auch dieses auf den alten Stand gebracht.
Wenn das alles erfolgt ist, hat man meiner Meinung nach den Spectrum mit der besten Tastatur und den besten Möglichkeiten. Sollte jemand auf der Tastatur die alten Keywords vermissen, so können sowohl Fred Dürkes (Clubleitung SUC) als auch ich eine Klebefolie anbieten, welche auf die Tasten geklebt, alle Keywords wieder erscheinen läßt (Meine sind zwar teurer, aber in Farbe).
Ich möchte hier noch anmerken, daß ein anderes Clubmitglied vom Computerflohmarkt mal einen Berg (ca 25) +2A/B aufkaufen konnte, welche ganz offensichtlich zwar als defekt angeboten wurden, aber wohl von unwissenden Usern benutzt worden waren. Diese Rechner laufen alle einwandfrei, so daß nur ein Grund zur Zurücksendung in Frage kommt: Die Leute haben Interface oder normalen Joystick angeschlossen, und waren über die Reaktion verblüfft.
Mich persönlich wundert, daß keiner der Rechner defekt war, denn wenn man ein Plus D z.B. anschließt, kann durchaus die ULA über die Wupper gehen, wie man hier sagt.
Und gerade die ULA dieser Rechner ist zwar im normalen Betrieb ein Dauerarbeiter, aber wenn Sie erst mal geschossen ist, ist der Ersatz fast nicht möglich.
Also möchte ich mal zusammenfassen:
Wer einen der genannten Rechner hat, sollte sich nicht scheuen, diesen umbauen zu lassen (oder es mit den Clubinfos selbstmachen), weil der Rechner erst dadurch wirklich brauchbar wird.

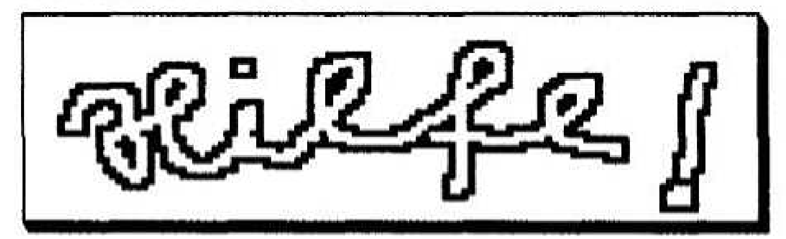
Peter Rennefeld, Genhodder 19
41179 Mönchengladbach, Tel. 02161/571141



Zu Mike Mee (Info 86, Seite 16)

Hallo Clubmitglieder!
Hier noch ein weiterer Kurzbericht über meine Kontakte zu Mike Mee in England, der alles sammeln wollte, was irgendwie mit dem Spectrum zusammenhängt, um dies dann auf eine CDROM zu brennen.
Leider hatte er sich ja über 2 Monate in keiner Weise bei mir gemeldet, so daß ich dachte, er wollte nicht. Am 18.02.97 erhielt ich jedoch tatsächlich einen Brief von ihm. Er beteuert, daß er wohl wegen Arbeitslosigkeit seinen PC verkaufen mußte, und deshalb nichts machen konnte. Auch seine Emails an mich (über eine Freundin von ihm) seien wohl auf seltsame Weise irgendwo verloren gegangen. (Angeblich wegen eines Registry-Fehlers in Windoof95, da lobe ich mir doch den Speccie).
Jedenfalls hat er wieder Arbeit, und will sich alsbald wieder einen PC zulegen. Vielleicht verfolgt er dann auch sein Projekt weiter ...
Mit spektralen Grüßen, Luzie

ps. Unter meiner Homepage auf
http://home.t-online.de/home/luzie/german.htm
könnt ihr immer eine aktualisierte File-Liste meiner ZX SpectROM laden, sowie auch die deutsche Übersetzung des TECHINFO.DOC vom Z80 Emulator v3.05.



Ich benötige dringend ein Handbuch für folgenden Drucker:
OKI Microline 182 Elite, Modell GE 5250 M.
Wer hat schon mal mit diesem Drucker gearbeitet und dadurch Erfahrung mit ihm am Spectrum? Ich bin für jede Hilfe dankbar. Unkosten z.B. durch Kopieren werden natürlich erstattet.

Lothar Ebelshäuser, Grasegger Straße 49
50737 Köln, Tel. 0221/747063